

<! --Topicos Especiales y Avanzados-->

<! --Ing. Amy Diaz-->

# Documentación de API {

<Por="Karen Lopez"/>

<Por="Junior Garcia"/>

}



# Introducción {

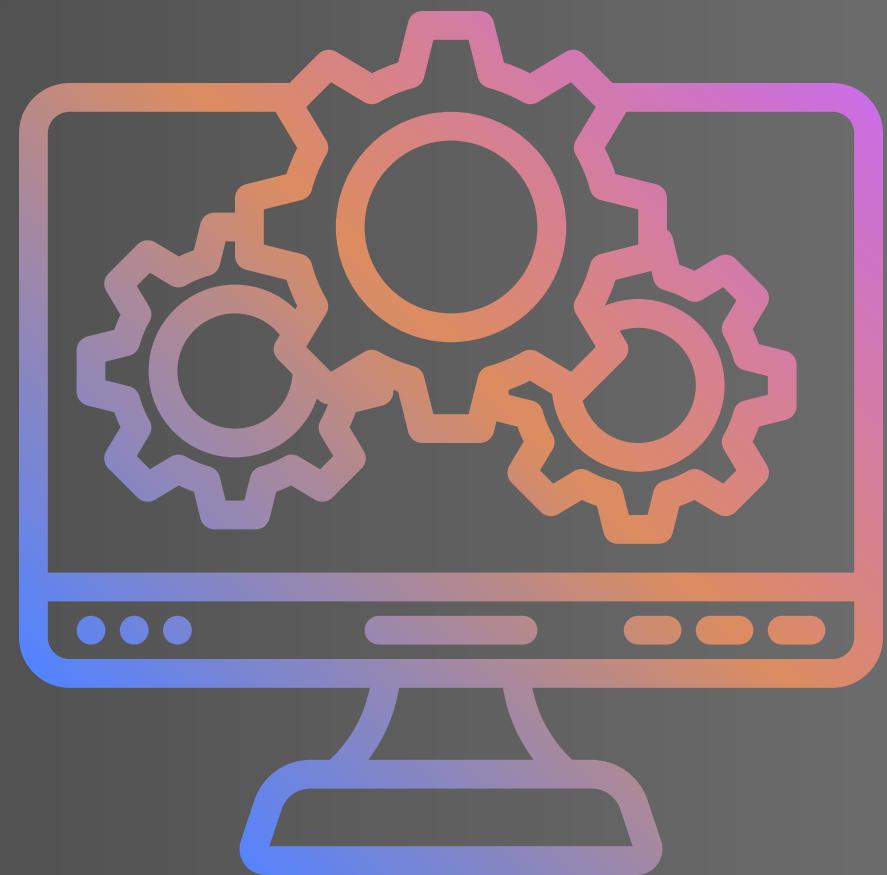
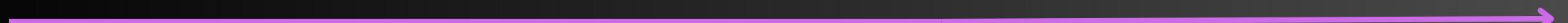
¿Que es un API?

Interfaz de  
Programación de  
Aplicaciones

}



# Beneficios y ejemplos de uso APIs

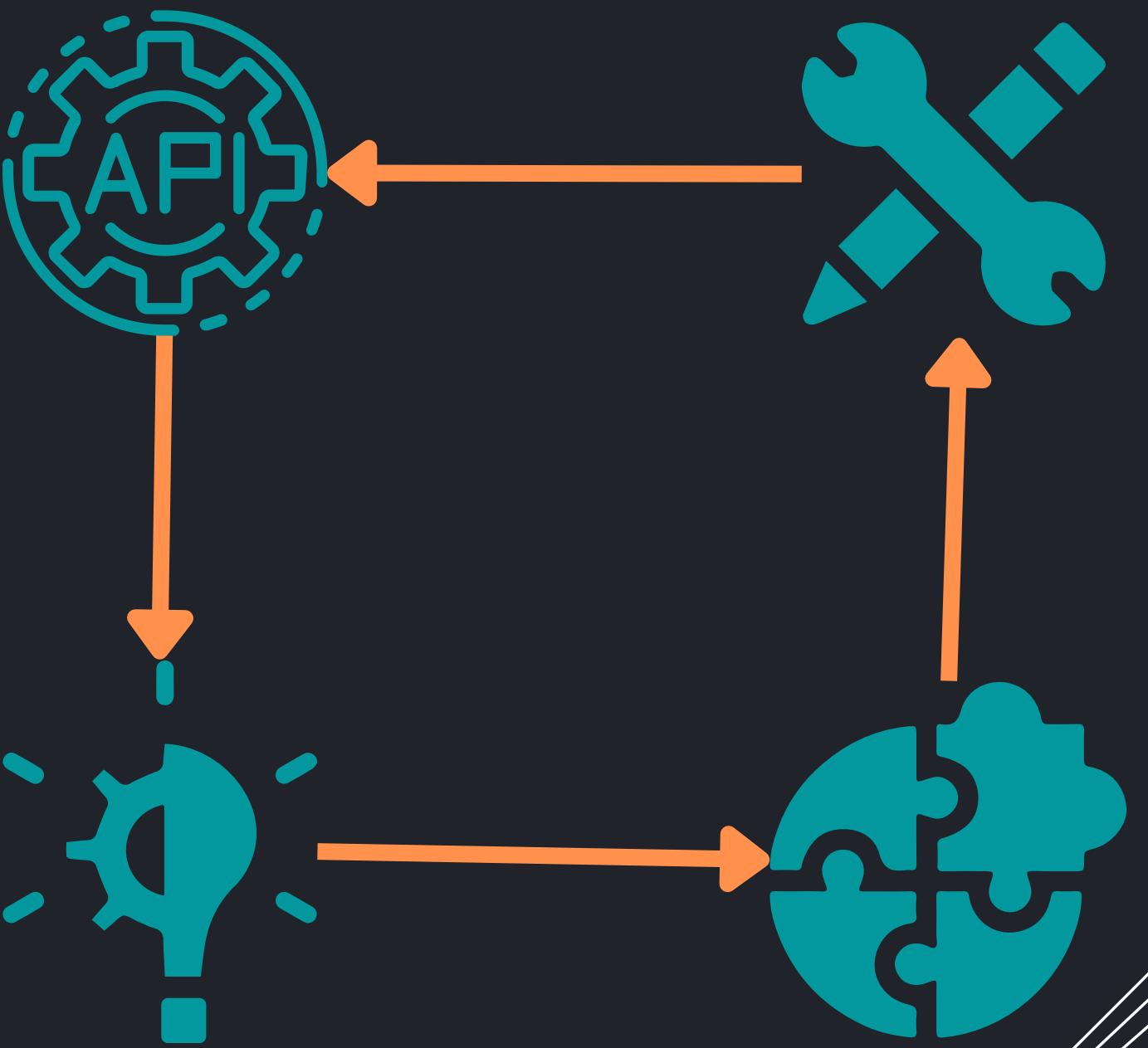


# Beneficios de API

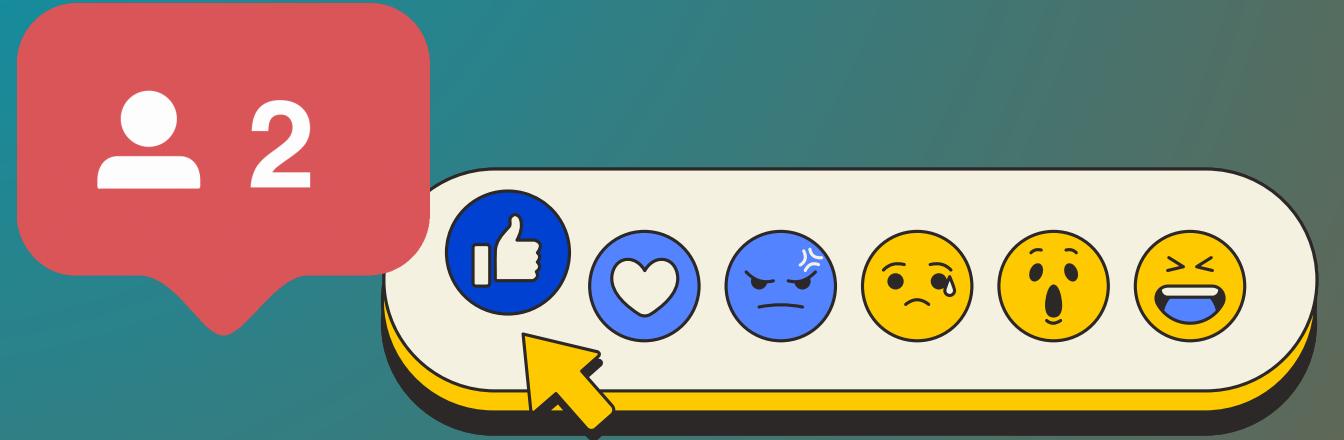
REST {

- Integración
- Innovación
- Ampliación
- Facilidad de mantenimiento

}



# Ejemplos de uso cotidiano



APIs de redes sociales

APIs de pasarelas de pago

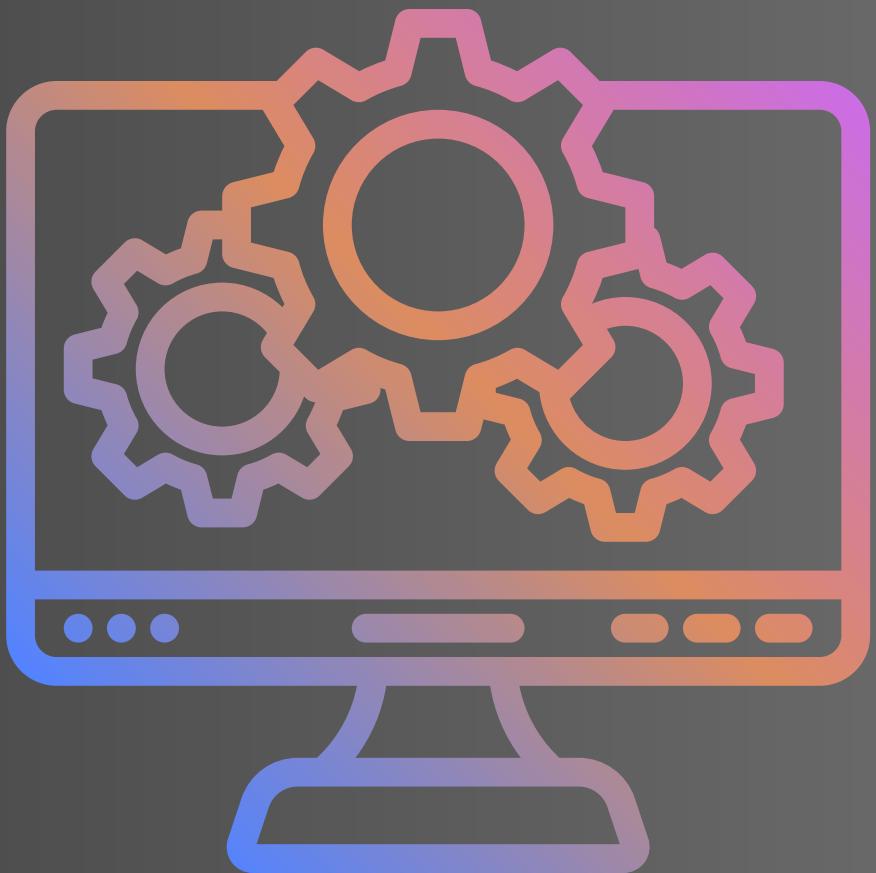


APIs de servicios en la nube

# Ciclo de vida de las APIs



# Tipos de Documentación de API

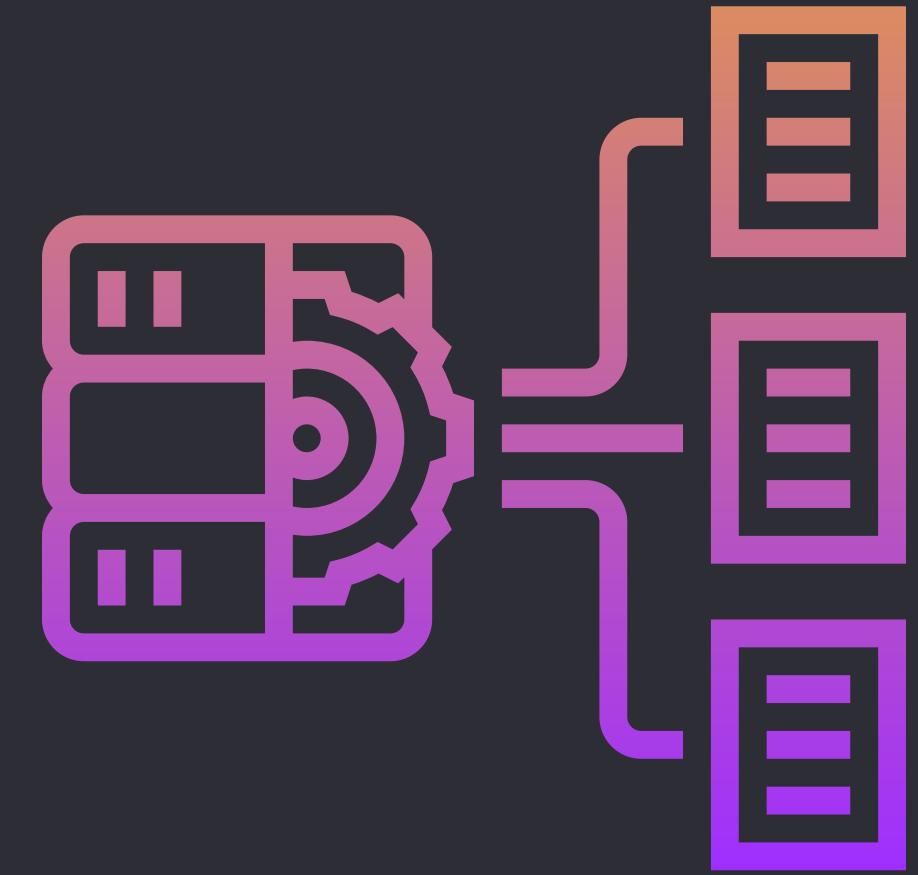
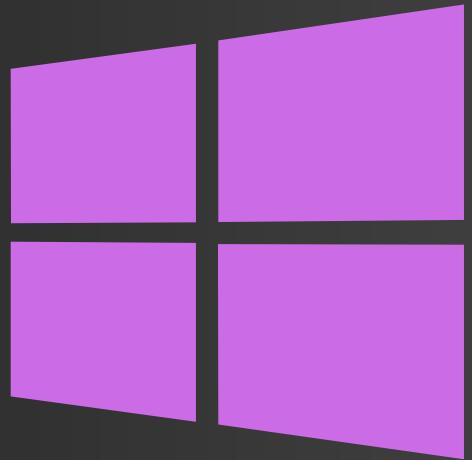


# Documentación de referencia

Proporciona información técnica detallada sobre todos los endpoints, métodos, parámetros, tipos de datos, códigos de estado y cualquier otro aspecto técnico de la API.



# Ejemplos de documentación de referencia

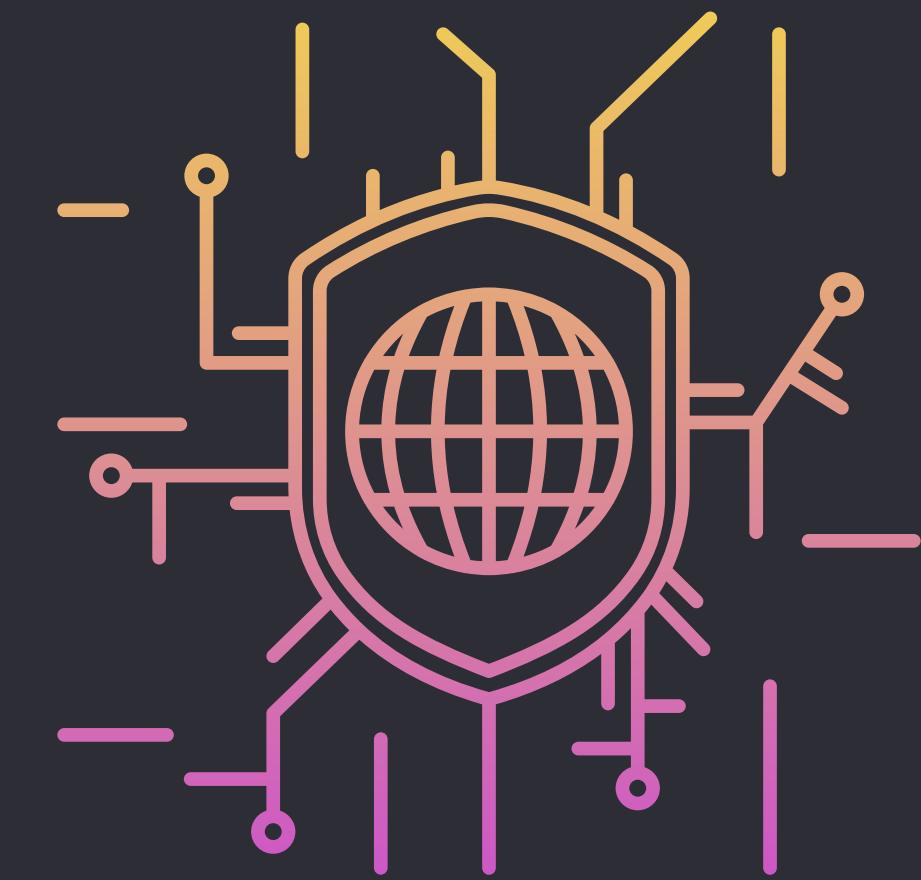
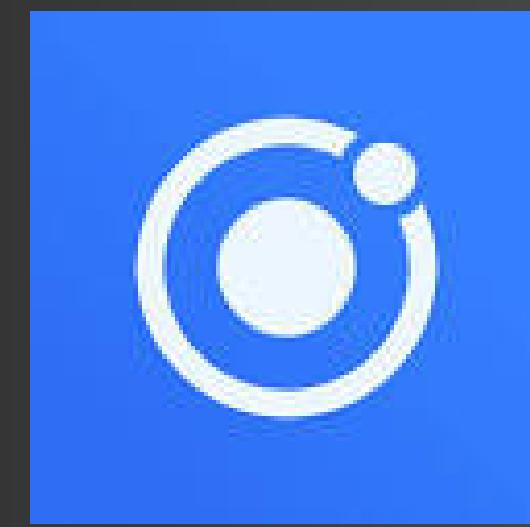
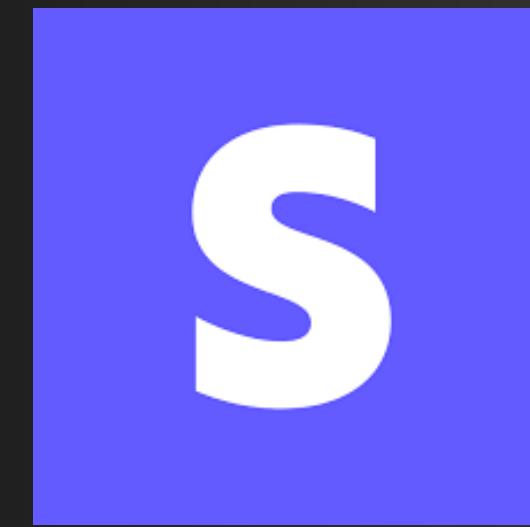


# Documentación de uso

Se centra en cómo utilizar la API de una manera más general. Incluye ejemplos y casos de uso que ayudan a los desarrolladores a comprender cómo aprovechar al máximo la API

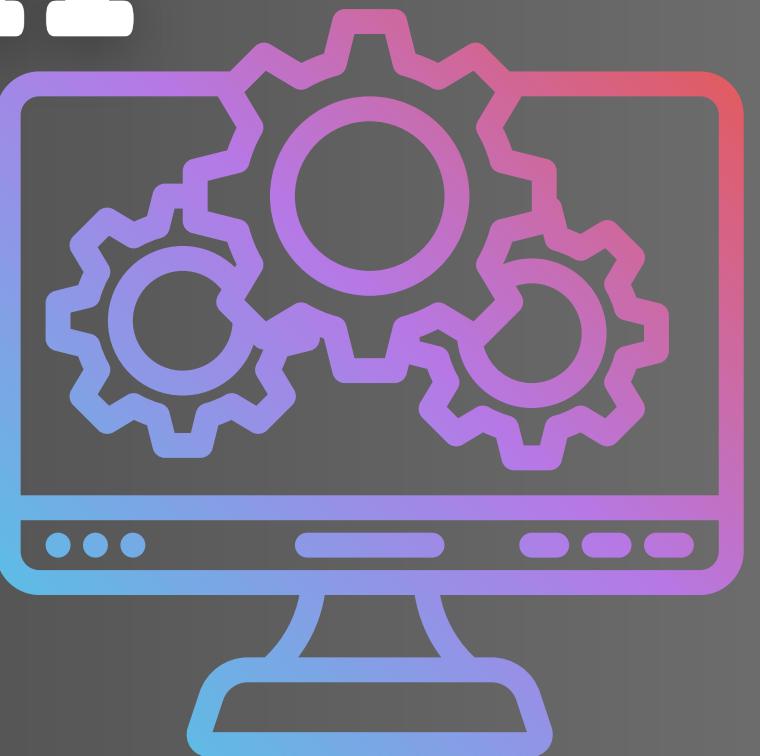


# Ejemplos de documentación de uso

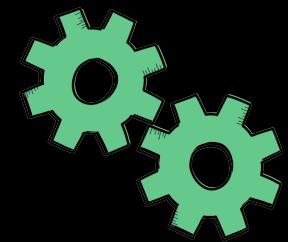


# Importancia de una Documentación

---



Si tu API no tiene documentación, nadie  
usará tu API en absoluto

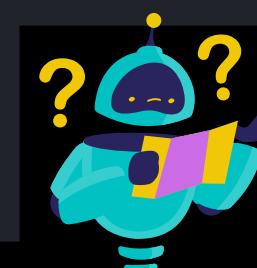


Una buena documentación ahorrará tiempo

en entender cómo funciona el API



Reducirá el número de preguntas  
innecesarias que tendría que responder  
el equipo de desarrollo



# Errores de documentación

01 - No subirlos al repositorio

02 - No actualizar documentos/código

03 - No tener orden en el versionamiento

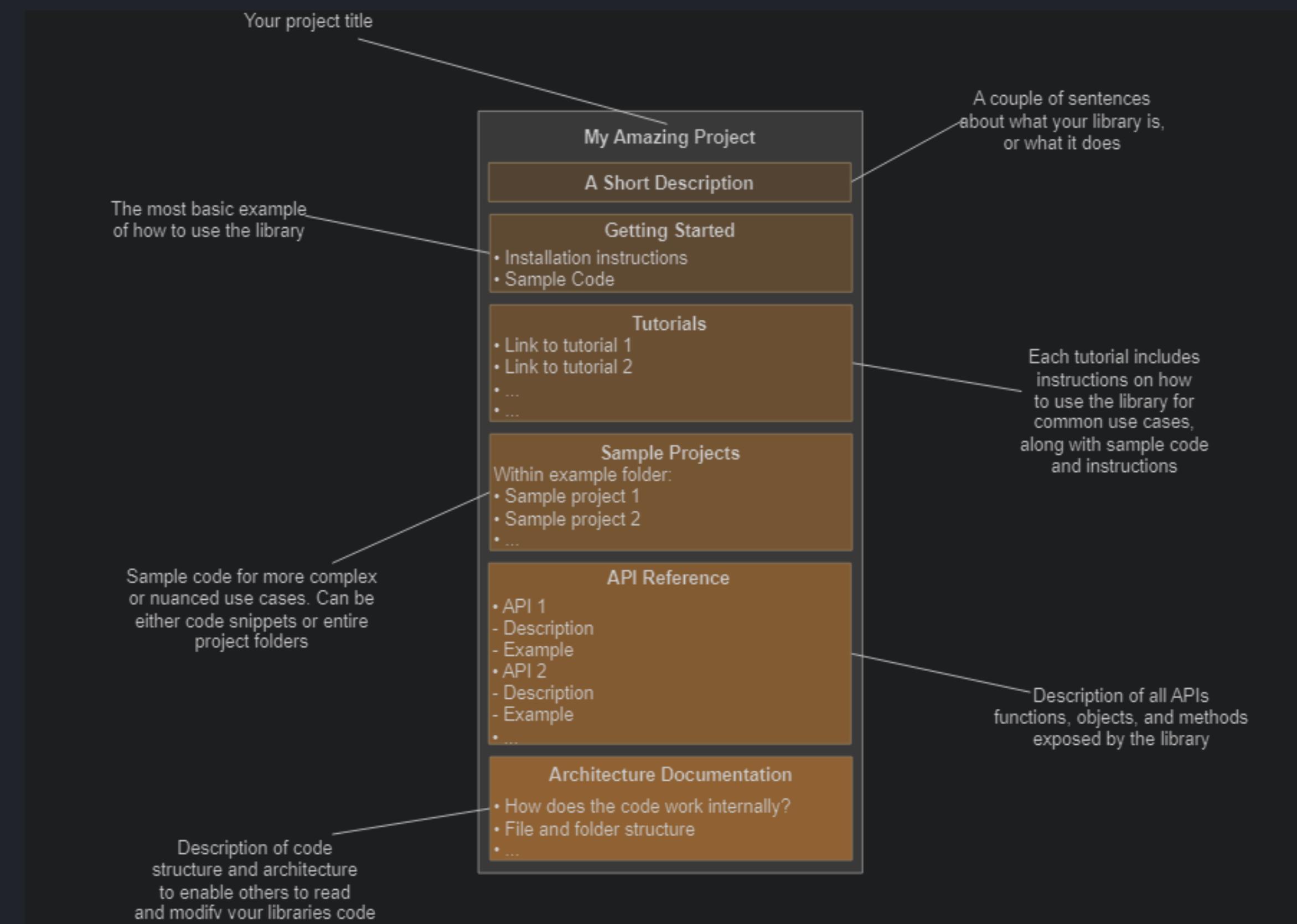
04 - No tener orden en las carpetas (arq, dev, qa)

05 - No seguir el estandar (plantillas)

# Elementos clave de una buena documentación



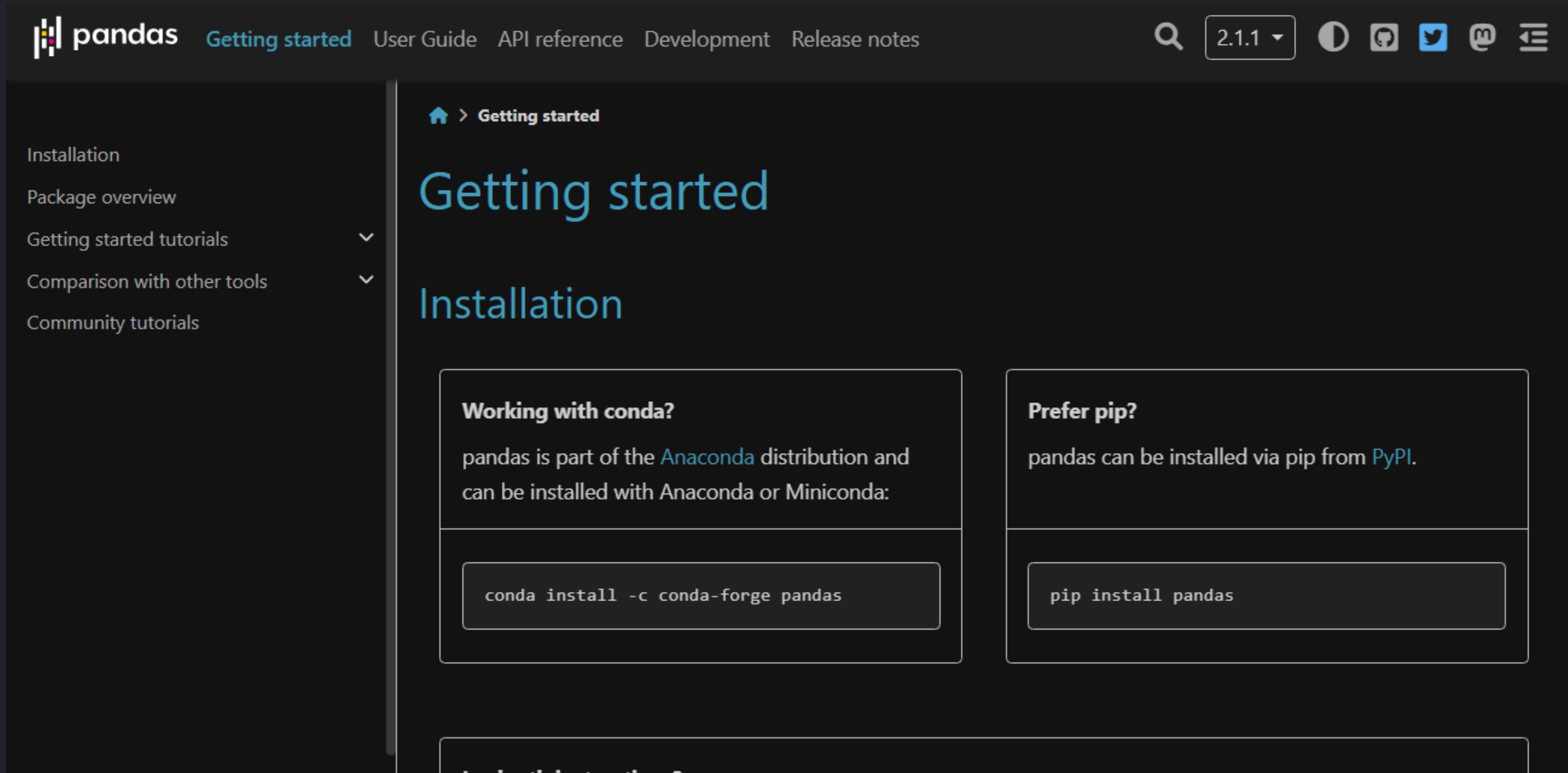
# Elementos clave de una buena documentación {



# Titulo del proyecto y breve descripción



# Guia de inicio rapido



The screenshot shows the "Getting started" section of the pandas documentation. The page has a dark background with light-colored text and code snippets. On the left, there's a sidebar with links like "Installation", "Package overview", "Getting started tutorials", "Comparison with other tools", and "Community tutorials". The main content area has a breadcrumb navigation bar at the top showing "Home > Getting started". The title "Getting started" is in large blue text, followed by "Installation" in blue. Below that, there are two sections: "Working with conda?" and "Prefer pip?". Each section contains a snippet of terminal command. At the bottom, there's a "Next" button.

**pandas** Getting started User Guide API reference Development Release notes

Getting started

Installation

Package overview

Getting started tutorials

Comparison with other tools

Community tutorials

Getting started

Installation

**Working with conda?**

pandas is part of the [Anaconda](#) distribution and can be installed with Anaconda or Miniconda:

```
conda install -c conda-forge pandas
```

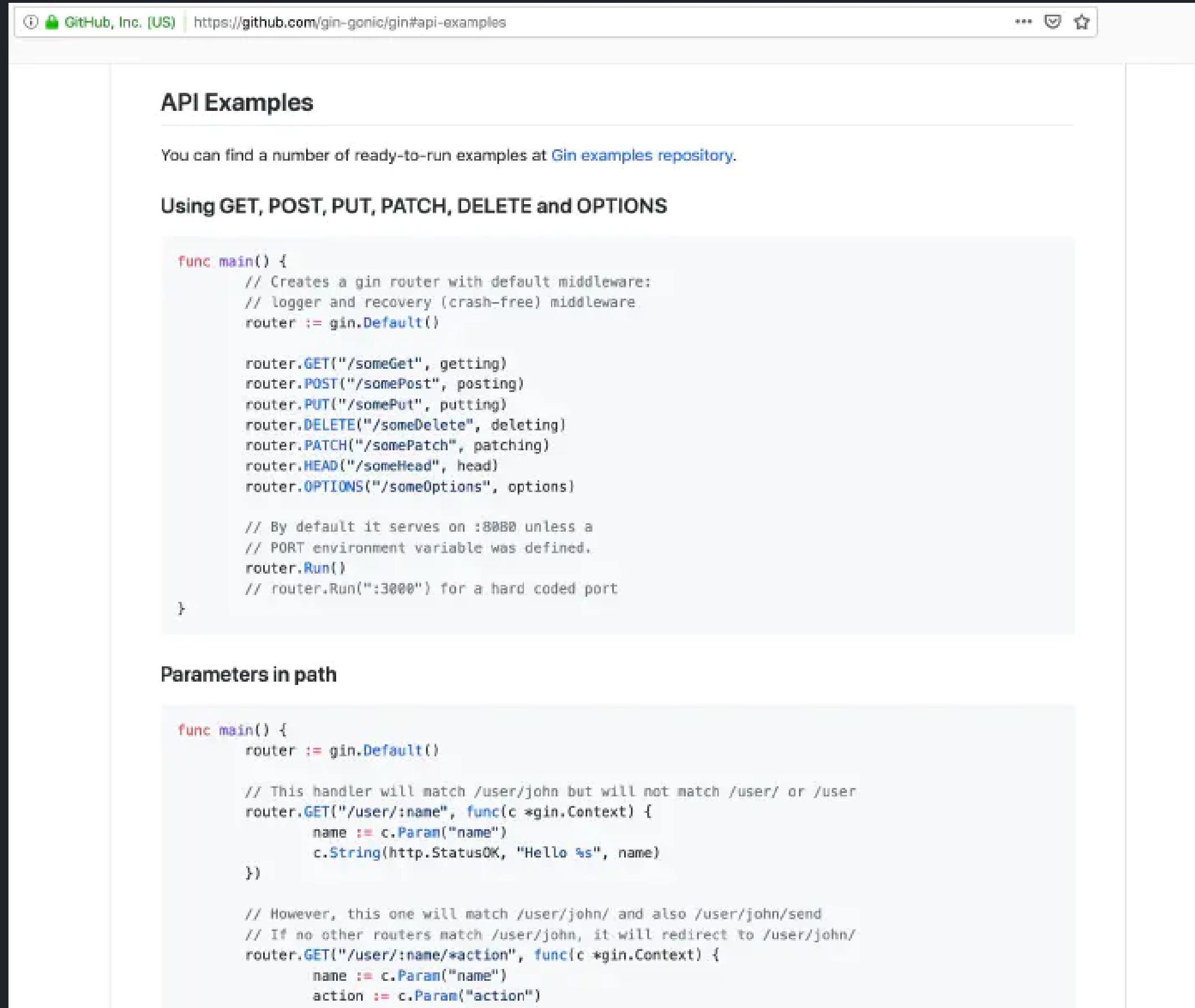
**Prefer pip?**

pandas can be installed via pip from [PyPI](#).

```
pip install pandas
```

Next

# Tutoriales



The screenshot shows a GitHub page for the 'gin-gonic/gin' repository, specifically the 'api-examples' branch. The title 'API Examples' is at the top. Below it, a note says 'You can find a number of ready-to-run examples at [Gin examples repository](#)'. A section titled 'Using GET, POST, PUT, PATCH, DELETE and OPTIONS' contains a code snippet:

```
func main() {
    // Creates a gin router with default middleware:
    // logger and recovery (crash-free) middleware
    router := gin.Default()

    router.GET("/someGet", getting)
    router.POST("/somePost", posting)
    router.PUT("/somePut", putting)
    router.DELETE("/someDelete", deleting)
    router.PATCH("/somePatch", patching)
    router.HEAD("/someHead", head)
    router.OPTIONS("/someOptions", options)

    // By default it serves on :8080 unless a
    // PORT environment variable was defined.
    router.Run()
    // router.Run(":3000") for a hard coded port
}
```

Below this, a section titled 'Parameters in path' contains another code snippet:

```
func main() {
    router := gin.Default()

    // This handler will match /user/john but will not match /user/ or /user/
    router.GET("/user/:name", func(c *gin.Context) {
        name := c.Param("name")
        c.String(http.StatusOK, "Hello %s", name)
    })

    // However, this one will match /user/john/ and also /user/john/send
    // If no other routers match /user/john, it will redirect to /user/john/
    router.GET("/user/:name/:action", func(c *gin.Context) {
        name := c.Param("name")
        action := c.Param("action")
    })
}
```

# Proyectos de muestra

The screenshot shows the GitHub repository page for `gin-gonic/examples`. The repository has 112 commits, 1 branch, 0 releases, 28 contributors, and is licensed under MIT. The latest commit was made 26 days ago by `appleboy` for a fix related to `bookableDate`. The commit history lists various contributions from different authors, including `app-engine`, `assets-in-binary`, `auto-tls`, `basic`, `custom-validation`, `favicon`, `file-binding`, `graceful-shutdown`, `grpc`, `http-pusher`, `http2`, `multiple-service`, `new_relic`, `realtime-advanced`, `realtime-chat`, `struct-lvl-validations`, `template`, and `upload-file`. The commits range from 2 years ago to last month.

Author	Commit Message	Date
<code>appleboy</code>	bookableDate fix (#20)	26 days ago
<code>app-engine</code>	correct repo url in README	6 months ago
<code>assets-in-binary</code>	Add example to build single binary with templates (#1328)	2 years ago
<code>auto-tls</code>	add package for govendor (#1166)	2 years ago
<code>basic</code>	chore: add some annotations (#1544)	last year
<code>custom-validation</code>	bookableDate fix	last month
<code>favicon</code>	chore: use http.Status* instead of hard code (#1482)	last year
<code>file-binding</code>	enhancement #8: add file binding example (#10)	6 months ago
<code>graceful-shutdown</code>	Fix typo (#21)	26 days ago
<code>grpc</code>	change github.com/gin-gonic/gin/examples/grpc/pb to github.com/gin-go...	8 months ago
<code>http-pusher</code>	Add Pusher() function for support http2 server push (#1273)	last year
<code>http2</code>	chore: use http.Status* instead of hard code (#1482)	last year
<code>multiple-service</code>	feat: add multiple service example. (#1119)	2 years ago
<code>new_relic</code>	Add NewRelic middleware example. (#1526)	8 months ago
<code>realtime-advanced</code>	Change the 'doctype' to be lowercase to be consistent with the rest o...	last year
<code>realtime-chat</code>	sync gin #382	8 months ago
<code>struct-lvl-validations</code>	fix(binding): Expose validator engine used by the default Validator (...)	2 years ago
<code>template</code>	fix file path	7 months ago
<code>upload-file</code>	Fix #1693: file.Filename should not be trusted. (#1699)	10 months ago

# Documentación de arquitectura

The screenshot shows the React documentation website with a dark theme. The top navigation bar includes a search icon, a search input field, and buttons for 'Ctrl K', 'Learn' (which is highlighted in blue), 'Reference', 'Community', 'Blog', and settings. On the left, a sidebar titled 'GET STARTED' has a dropdown menu for 'Quick Start' which is currently expanded. Under 'Quick Start', there's a link to 'Tutorial: Tic-Tac-Toe'. Below this, the 'Thinking in React' section is highlighted with a blue background. Other items in the sidebar include 'Installation', 'LEARN REACT', 'Describing the UI', 'Adding Interactivity', 'Managing State', and 'Escape Hatches'. The main content area is titled 'Thinking in React' and discusses the components-based approach to building user interfaces. It mentions breaking down designs into components, describing visual states, and connecting them. A sub-section titled 'Start with the mockup' is shown, along with JSON API data examples.

React can change how you think about the designs you look at and the apps you build. When you build a user interface with React, you will first break it apart into pieces called **components**. Then, you will describe the different visual states for each of your components. Finally, you will connect your components together so that the data flows through them. In this tutorial, we'll guide you through the thought process of building a searchable product data table with React.

## Start with the mockup

Imagine that you already have a JSON API and a mockup from a designer.

The JSON API returns some data that looks like this:

```
[ { category: "Fruits", price: "$1", stocked: true, name: "Apple" },
```

Thinking\_in React

# Rest API {

**[Título API]**

**[Descripción]**

`http://<SERVER_ADDRESS>/`

**Método:** GET/PUT/POST/DELETE

**Authentication:** basic authentication/API key/OAuth

**Request Parameters**

Header fields

Header Field	Value
Authorization	A valid access token

**Parameters**

Parameter	Type	Description
id	string	User ID <b>REQUIRED</b>

1 TITULO/INTRODUCCION

2 ENDPOINT

3 METODO

4 AUTENTICACION

5 PARAMETROS

## 6 JSON REQUEST BODY

### JSON Request Body

Value	Type	Description	
serialNumber	string	<i>Serial Number</i>	REQUIRED

### HTTP Response

HTTP Code	Description
200	Successful Response
401	Unauthorized

### Error or Warning Response

invalid value for `field\_name`

Unrecognized values for parameter paramname: value1, value2, value3

## 7 RESPONSE

## 8 EJEMPLO

Plantilla -> { }

# Mejores prácticas en la Documentación





# Mejores prácticas en la Documentación de API {



## Planificar

Tener un plan de acción para elaborar la documentación

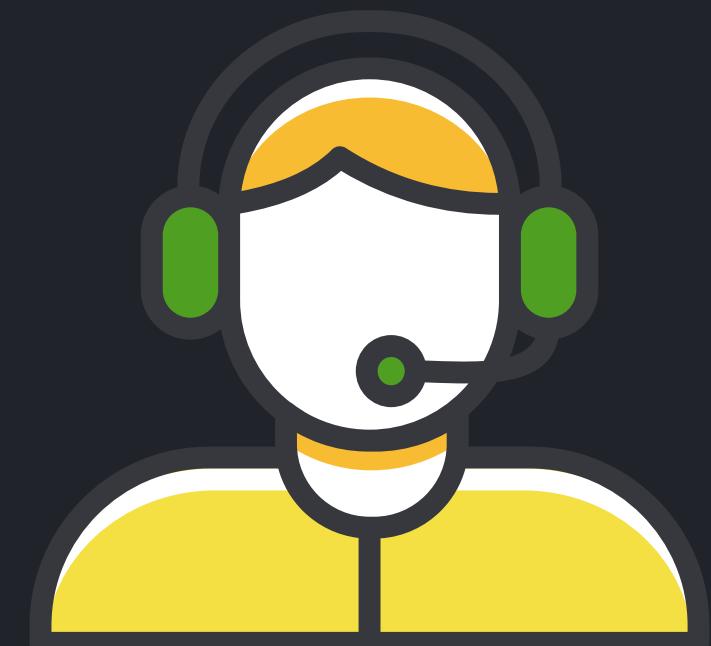
## Actualizar la documentación

Asegurar que la documentación sea actualizada periódicamente



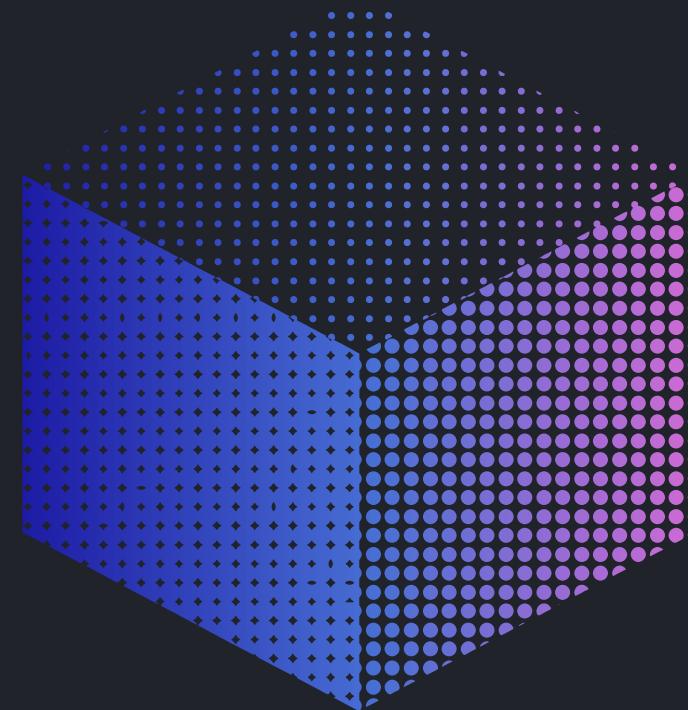
## Responsable

Encargado de desarrollar y mantener la documentación



## Pensar en la audiencia

Escribir la documentación pensando en la audiencia que la usara

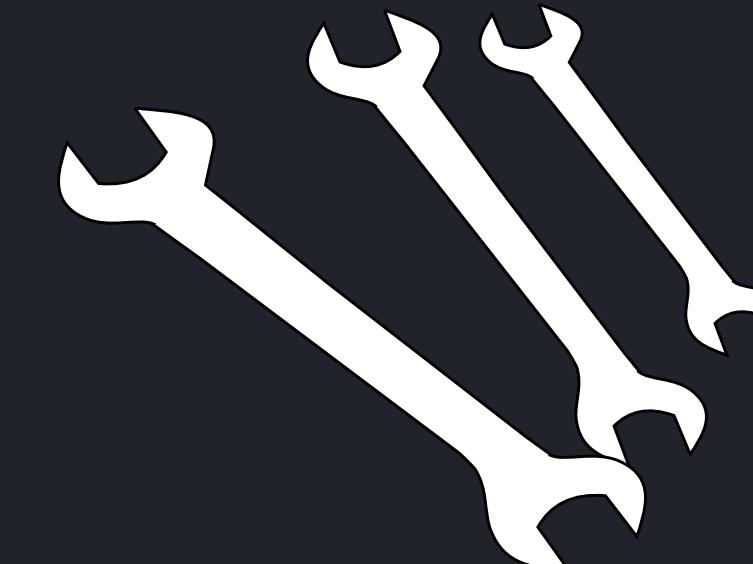


## Consistencia y evitar jerga

Documentación  
uniforme y con  
la misma  
terminología y  
nomenclatura

## Crear ejemplos

Incluir código  
funcional y que  
pueda ser testeado  
de acuerdo  
a la documentación  
sugerida

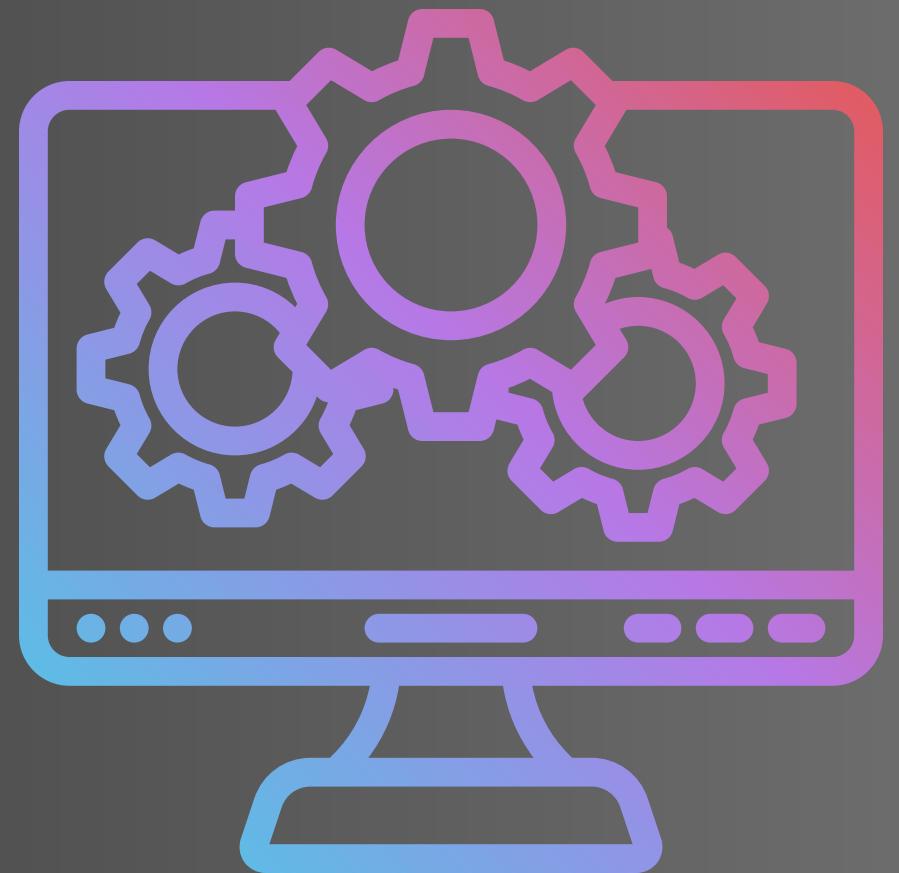


## Usar herramientas

Permiten facilitar  
el trabajo y ayudan  
a crear la  
documentación de  
forma dinámica

}

# Herramientas de Documentación de API



# HERRAMIENTAS{

## Swagger

The screenshot shows the Swaggerhub interface. On the left, there is a code editor window titled "swagger-hub... | registry-a... | 1.0.45" containing a Swagger JSON specification. The specification includes details like the swagger version (2.0), info (description, introduction, authentication instructions), and various API definitions with their operations (GET, POST, PUT, DELETE) and descriptions. On the right, there is a list of "APIs: Operations for APIs" with corresponding HTTP methods and URLs. The interface has a dark theme with some UI elements in green and red.

```
swagger: '2.0'
info:
  description: |
    # Introduction
    This is the registry API for SwaggerHub. It allows you to access, manage, and update your APIs and Domains in SwaggerHub bypassing the Web application.

  # Authentication
  Use your personal API Key: you may find it by visiting the [API Key page](https://app.swaggerhub.com/settings/apiKey).

version: 1.0.45
title: SwaggerHub Registry API
contact:
  name: SwaggerHub
  url: 'http://swaggerhub.com'
  email: info@swaggerhub.com
host: api.swaggerhub.com
tags:
  - name: APIs
    description: Operations for APIs
  - name: Domains
    description: Operations for Domains
schemes:
  - https
produces:
  - application/json
paths:
  /specs:
    get:
      tags:
        - APIs
        - Domains
      summary: >-
        Retrieves a list of currently defined APIs and Domains in APIs.json
      format:
      descriptions: ''
      operationId: searchApisAndDomains
      parameters:
        - name: specType
          in: query
          description: |
            Type of Swagger spec to search
```

Operations	HTTP Method	URL	Description	Status
Retrieves a list of currently defined APIs and Domains in APIs.json	GET	/specs	Retrieves a list of currently defined APIs and Domains in APIs.json format.	VALID
Retrieves a list of currently defined APIs	GET	/apis	Retrieves a list of currently defined APIs in APIs.json format.	PUBLISHED
Retrieves an APIs.json listing of all APIs defined for this owner	GET	/apis/{owner}	Retrieves an APIs.json listing of all APIs defined for this owner	PUBLISHED
Retrieves an APIs.json listing for all API versions for this owner and API	GET	/apis/{owner}/{api}	Retrieves an APIs.json listing for all API versions for this owner and API	PUBLISHED
Saves the provided Swagger definition	POST	/apis/{owner}/{api}	Saves the provided Swagger definition	PUBLISHED
Deletes the specified API	DELETE	/apis/{owner}/{api}	Deletes the specified API	PUBLISHED
Gets API's collaboration	GET	/apis/{owner}/{api}/.collaboration	Gets API's collaboration	PUBLISHED
Updates API's collaboration	PUT	/apis/{owner}/{api}/.collaboration	Updates API's collaboration	PUBLISHED
Deletes API's collaboration	DELETE	/apis/{owner}/{api}/.collaboration	Deletes API's collaboration	PUBLISHED
Retrieves the Swagger definition for the specified API and version	GET	/apis/{owner}/{api}/{version}	Retrieves the Swagger definition for the specified API and version	PUBLISHED
Deletes a particular version of the specified API	DELETE	/apis/{owner}/{api}/{version}	Deletes a particular version of the specified API	PUBLISHED

# Postman

The screenshot shows the Postman API documentation for the `POST /collections` endpoint. The left sidebar lists various collections, environments, mocks, monitors, and workspaces. The main content area is titled "POST Create Collection" and provides details about creating collections using the Postman Collection v2 format. It includes an example request in JavaScript and an example response in JSON.

**Example Request:**

```
var https = require('https');

var options = {
  'method': 'POST',
  'hostname': 'api.getpostman.com',
  'path': '/collections',
  'headers': {
    'X-Api-Key': '{{postman_api_key}}',
    'Content-Type': 'application/json'
  }
}
```

**Example Response:**

```
200 – OK
```

```
{
  "collection": {
    "id": "2412a72c-1d8e-491b-aced-93809c0e94e9",
    "name": "Sample Collection",
    "uid": "5852-2412a72c-1d8e-491b-aced-93809c0e94e9"
  }
}
```

# Apiary

The screenshot displays the Apiary API Blueprint Editor interface, specifically the Notes API documentation. The left side shows the API Blueprint code in a monospaced font, detailing various endpoints like listing notes, creating a note, retrieving a note, and removing a note. The right side provides a detailed view of the API, including the introduction, reference, and a request section with a mock server and JavaScript code.

**Notes API Blueprint Code:**

```
1 FORMAT: 1A
2 HOST: http://api.google.com
3
4 # Notes API
5 Notes API is a *short texts saving* service similar to its physical paper presence on your table.
6
7 # Group Notes
8 Notes related resources of the **Notes API**.
9
10 ## Notes Collection [/notes]
11 ### List all Notes [GET]
12 + Response 200 (application/json)
13
14 [
15   {
16     "id": 1, "title": "Jogging in park"
17   },
18   {
19     "id": 2, "title": "Pick-up posters from post-office"
20 }
21
22 ## Create a Note [POST]
23 + Request (application/json)
24
25   { "title": "Buy cheese and bread for breakfast." }
26
27 + Response 201 (application/json)
28
29 ## Note [/notes/{id}]
30 A single Note object with all its details.
31
32 + Parameters
33   + id (required, number, '1') ... Numeric 'id' of the Note to perform action on.
34
35 ## Retrieve a Note [GET]
36 + Response 200 (application/json)
37
38 + Header
39
40   X-My-Header: The Value
41
42 + Body
43
44   { "id": 2, "title": "Pick-up posters from post-office" }
45
46 ## Remove a Note [DELETE]
47 + Response 204
```

**Notes API Documentation:**

**INTRODUCTION**  
Notes API is a short texts saving service similar to its physical paper presence on your table.

**REFERENCE**  
Notes

**Notes**  
Notes related resources of the **Notes API**

**Notes Collection**

- List all Notes >
- Create a Note >

**Request**

Notes / Notes Collection / List all Notes

GET http://api.google.com/notes

Mock Server ▾ JavaScript ▾ Try

```
1 var Request = new XMLHttpRequest();
2
3 Request.open('GET', 'http://private-10ca8-notesapi152.apiary-mock.com/notes');
4
5 Request.onreadystatechange = function () {
6   if (this.readyState === 4) {
7     console.log('Status:', this.status);
8     console.log('Headers:', this.getAllResponseHeaders());
9     console.log('Body:', this.responseText);
10  }
11 }
12
13 Request.send(JSON.stringify(body));
```

}

# Ejemplo de Documentación de API

---





Swagger

TM



# Entidades que utilizan OPENAPI

ebay

IBM

Q Rapid

Microsoft

SAP®

apiplatform.io

IFS

Geek.Zone

CISCO

Google

```
SWAGGER {
```

Es un estándar que describe, produce, consume y visualiza REST API, APIs web services.

```
cout << "Enter rows and columns for second matrix." << endl;
cout << "Enter elements of first matrix." << endl;
cout << "Enter elements of matrix 1: " << endl;
for (int i = 0; i < m1; ++i)
    for (int j = 0; j < n1; ++j)
        cout << "Enter element a" << i + 1 << j + 1 << ":" ;
cout << endl;
cout << "Enter elements of second matrix." << endl;
cout << "Enter elements of matrix 2: " << endl;
for (int i = 0; i < m2; ++i)
    for (int j = 0; j < n2; ++j)
        cout << "Enter element b" << i + 1 << j + 1 << ":" ;
cout << endl;
```

PASOS{

Agregar  
dependencia  
en el pom.xml

```
<dependency>  
  
<groupId>com.newrelic.agent.java</groupId>  
    <artifactId>newrelic-api</artifactId>  
    <version>4.2.0</version>  
</dependency>
```

# PASOS{

Agregar los  
import

```
import  
com.newrelic.api.  
agent.Trace
```

En el método  
getrequest  
agregamos la  
URL de  
nuestra API:

```
@Trace(metricName =  
"mobile/tigo/hn/digital/  
customer/credit/assessme  
nt/v1/consumer/{consumer  
}", dispatcher = true)
```

Vamos a  
SwaggerEditor  
e importamos  
nuestro  
archivo :



```
https://editor.  
swagger.io/
```

}

# Conclusiones {

"La documentación es una carta de amor que le escribes a tu yo futuro". --Damian Conway

}

# Enlace de Presentacion

{



Documentación

}

```
<!--Grupo # 14 -->
```

Gracias {

<Por='tu atención'>

}